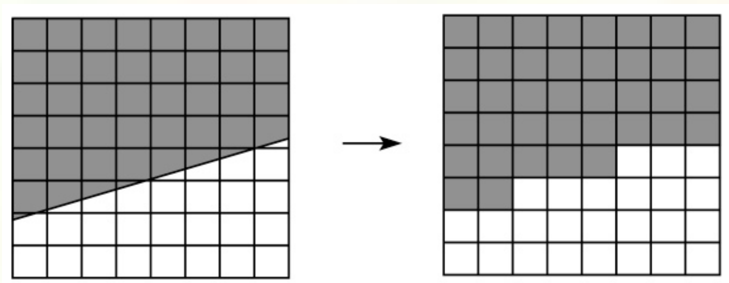
Anti-aliased and accelerated ray tracing

Reading

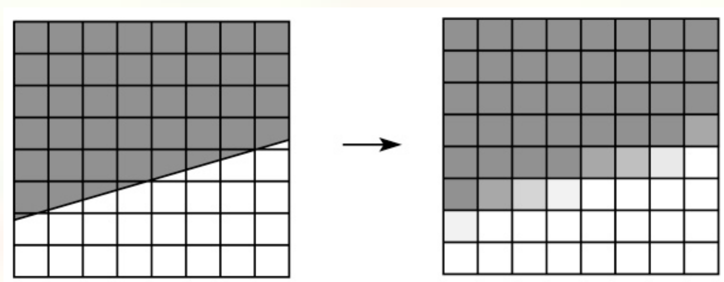
- Required:
  - Watt, sections 12.5.3 – 12.5.4, 14.7
- Further reading:
  - A. Glassner. An Introduction to Ray Tracing. Academic Press, 1989. [In the lab.]

Aliasing in rendering

- One of the most common rendering artifacts is the "jaggies". Consider rendering a white polygon against a black background:



- We would instead like to get a smoother transition:

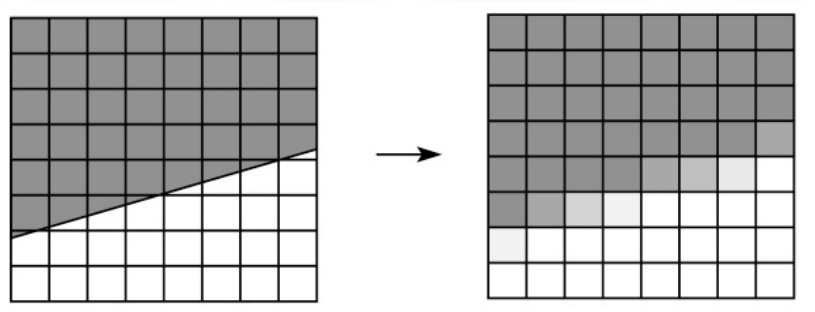


Anti-aliasing

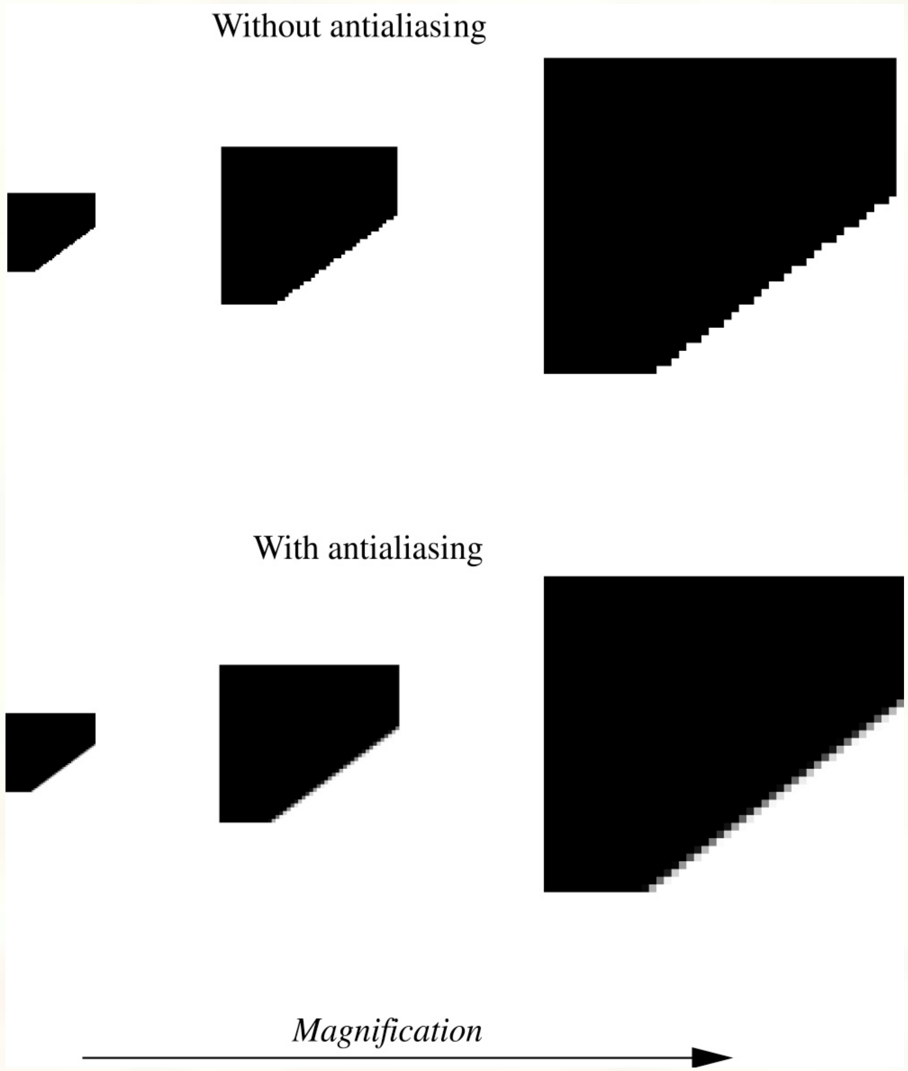
- **Q**: How do we avoid aliasing artifacts?

1. Sampling:
2. Pre-filtering:
3. Combination:

- Example - polygon:

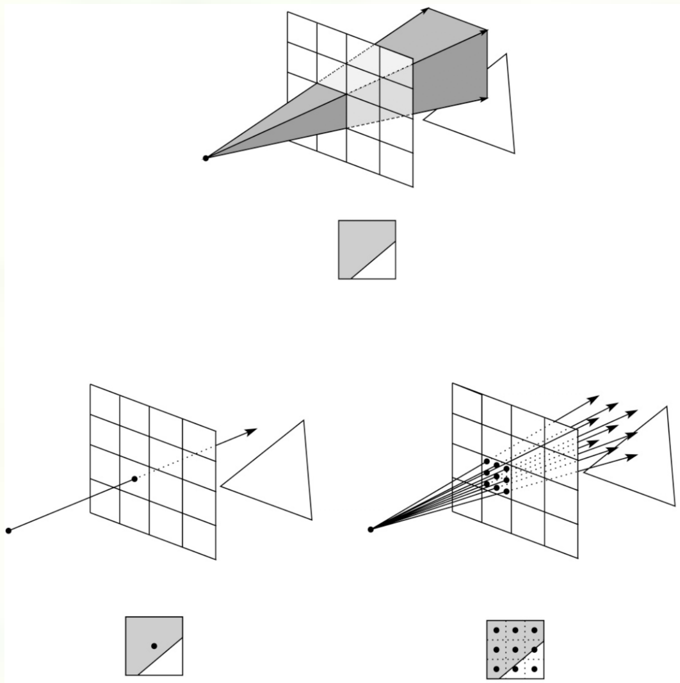


Polygon anti-aliasing



Antialiasing in a ray tracer

- We would like to compute the average intensity in the neighborhood of each pixel.



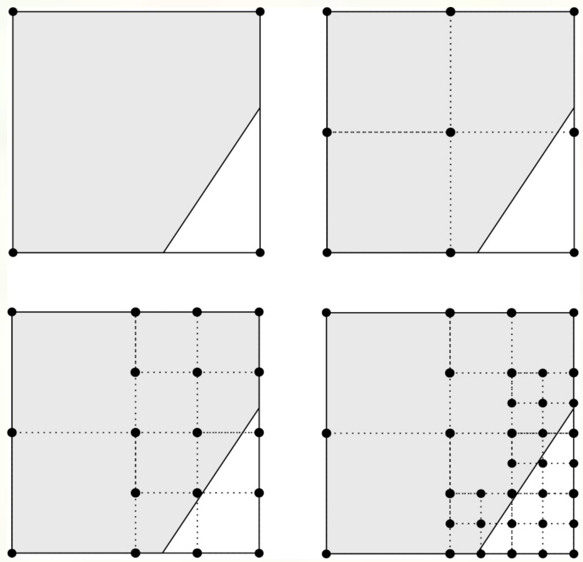
- When casting one ray per pixel, we are likely to have aliasing artifacts.
- To improve matters, we can cast more than one ray per pixel and average the result.
- A.k.a., **super-sampling and averaging down**.

Speeding it up

- Vanilla ray tracing is really slow!
- Consider: m x m pixels, k x k supersampling, and n primitives, average ray path length of d, with 2 rays cast recursively per intersection.
- Complexity =
- For m=1,000,000, k = 5, n = 100,000, d=8…very expensive!!
- In practice, some acceleration technique is almost always used.
- We've already looked at reducing d with adaptive ray termination.
- Now we look at reducing the effect of the k and n terms.

Antialiasing by adaptive sampling

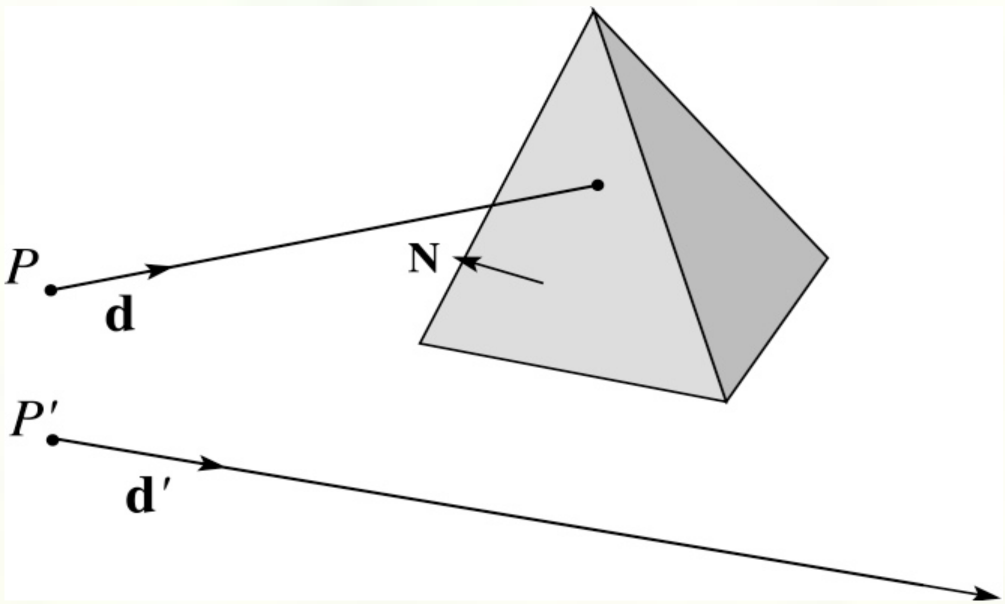
- Casting many rays per pixel can be unnecessarily costly.
- For example, if there are no rapid changes in intensity at the pixel, maybe only a few samples are needed.
- Solution: **adaptive sampling**.



- **Q**: When do we decide to cast more rays in a particular area?

Faster ray-polyhedron intersection

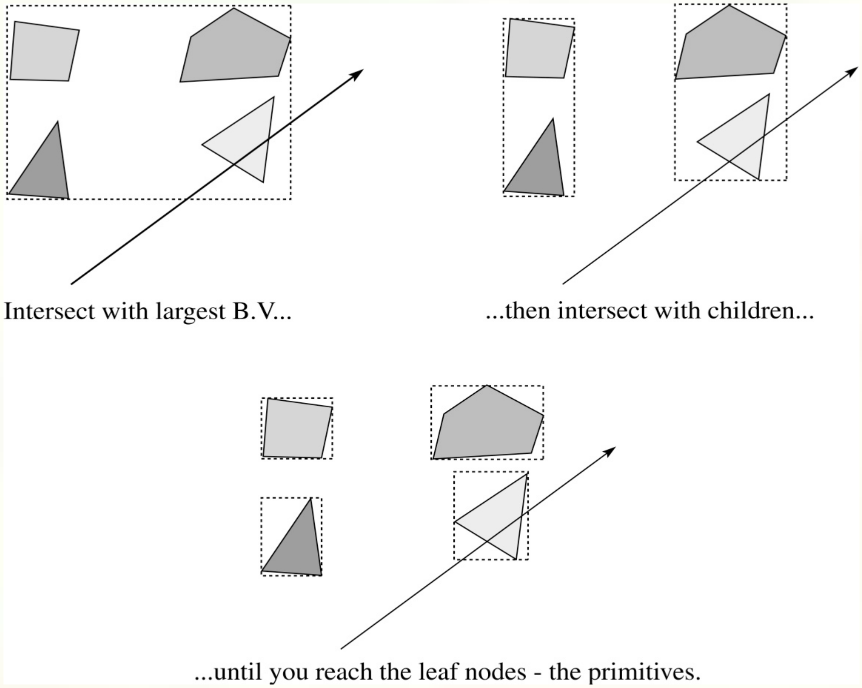
- Let's say you were intersecting a ray with a polyhedron:



- Straightforward method
  - intersect the ray with each triangle
  - return the intersection with the smallest *t*-value.
- **Q**: How might you speed this up?

Hierarchical bounding volumes

- We can generalize the idea of bounding volume acceleration with **hierarchical bounding volumes**.

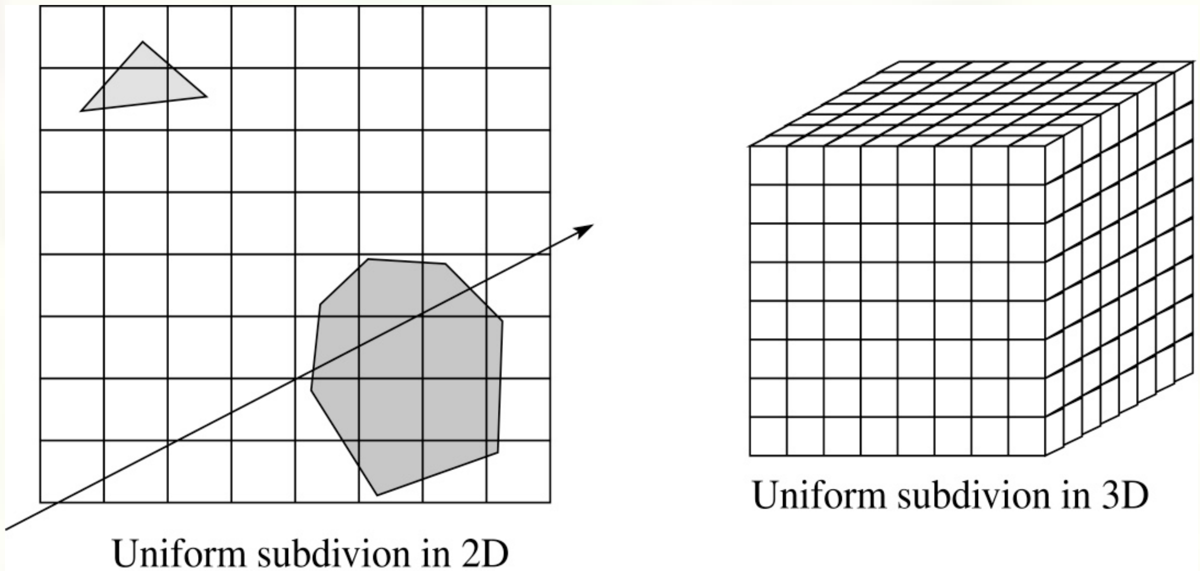


- Key: build balanced trees with *tight bounding volumes*.

  Many different kinds of bounding volumes.
  Note that bounding volumes can overlap.

Uniform spatial subdivision

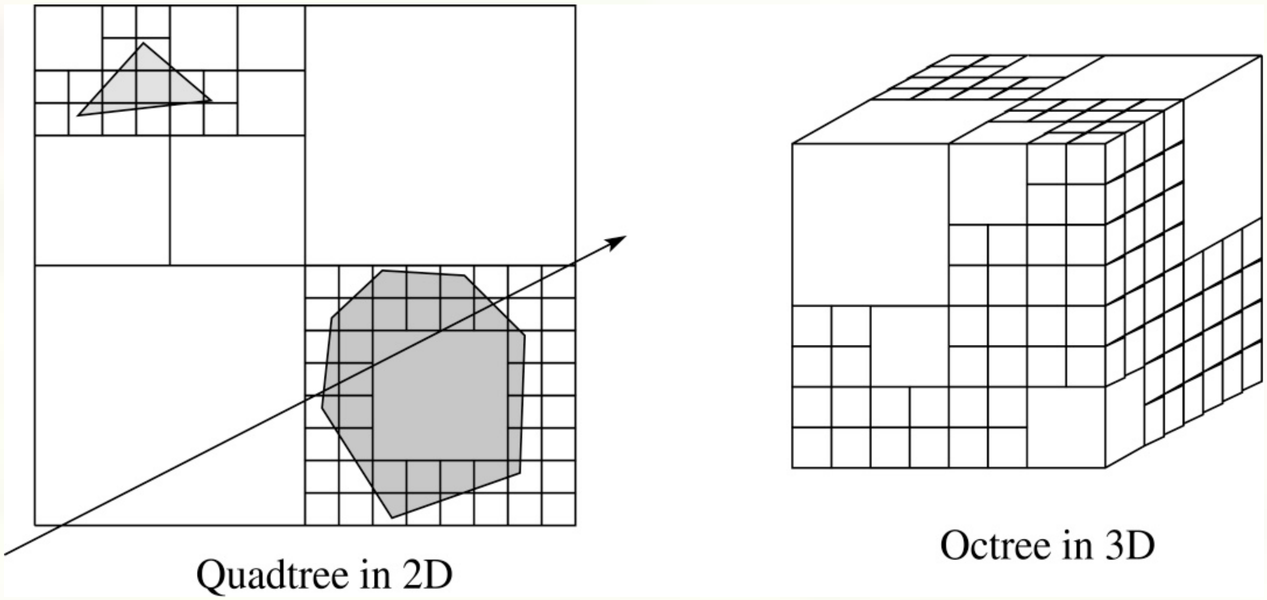
- Another approach is **uniform spatial subdivision**.



- Idea:
  - Partition space into cells (voxels)
  - Associate each primitive with the cells it overlaps
  - Trace ray through voxel array *using fast incremental arithmetic* to step from cell to cell

Non-uniform spatial subdivision

- Still another approach is **non-uniform spatial subdivision**.



- Other variants include k-d trees and BSP trees.

- Various combinations of these ray intersections techniques are also possible. See Glassner and pointers at bottom of project web page for more.